历史数据服务，重要考虑到因素

## 性能设计需求

历史数据存储量很大，插入、查询性能需要综合优化设计，我们需要支持的场景：

1. 每个节点最多10W个TAG点
2. 每个TAG点保存10年历史数据
3. 数据的粒度假定为：每个点每秒保存一个数据
4. 原始数据，需要保存1年

那么需要保存的原始数据的总量为：

10W点\*360天\*86400秒/天=3600W\*8.6400W=3.1104万亿条记录

## 表类型设计

考虑到记录条数和访问速度的平衡，我们设计三类表

1. 实时数据临时表：每条记录就是1个原始数据，但保存时间很短
2. 原始数据表：每条记录合并保存了若干个原始数据
3. 统计数据表：每条记录保存了一段时间内数据的最大、最小、第一个值，原始数据不保存
4. 实时数据临时表，表名为：temp

实时数据经常被趋势图组件访问，每个点每秒钟会访问一次，每次访问往往是取增量1-少数几个数据，具备以下特性：

* 总记录条数必须较少，加快检索速度
* 存放周期短，超过该周期的实时数据必须到raw\_yyyyMMdd去查询获取
* 每个数据存放一条记录，访问一次通讯量小
* 每来一条数据就更新一次，具备实时性

据此，设计临时temp表：

* 存放10分钟的原始数据，超过10分钟的数据，会自动转储到其他原始数据表和统计数据表。因此，必须对记录进行归并保存
* 最大记录量为：10W\*600=6000W条记录

记录示例：

{

"n": "Obj.Tag1",

"t": "2017-04-17 16:25:46.299",

"q": "0",

"v": "1.1"

}

1. 原始数据表，表名为：raw\_yyyyMMdd

原始数据表

* 每天一个数据表，表名规则：raw\_yyyyMMdd，如raw\_20150702
* 每10分钟才更新一次，所以不能当做实时数据进行访问
* 优化：每条记录保存10分钟数据，合并保存了一个点最多600个数据，大小可能有10K，所以该表不能作为实时表使用。如果10分钟内没有就，那么该10分钟内不保存数据
* 每个表最大数据量：10W点\*24\*6=1440万条记录
* 数据记录来自临时表，从临时表进行计算
* 启动程序时，需要考虑上次临时表temp中的数据，可能需要转存到原始数据表

表raw\_20170417记录示例：

{

"n": " Obj.Tag1",

"t": "2017-04-17 15:20",

"vs": [

{

"v": "1",

"t": "2017-04-17 16:25:46.299",

"q": "0"

},

{

"v": "1.2",

"t": "2017-04-17 16:25:47.299",

"q": "0"

}

]

}

1. 统计表-分钟表，表名为：minute\_yyyyMM

原始数据表

* 每个月一个数据表，表名规则：minute\_yyyyMM，如raw\_201507
* 每个数据表示1分钟内的最大、最小、初始值、时间、质量等属性
* 优化：每条记录保存每小时的60个数据，合并存放在1条记录中。
* 每1分钟才更新一次
* 记录量：10W\*30\*24=7200万条

表minute \_201704记录示例：

{

"n": "Obj.Tag1",

"t": "2017-04-17 15:20:25",

"vs": [

{

"t": "2017-04-17 16:25",

"fv": "1",

"ft": "2017-04-17 16:25:46.299",

"fq": "0",

"maxv": "1.2",

"maxt": "2017-04-17 16:25:47.299",

"maxq": "0",

"minv": "1.2",

"mint": "2017-04-17 16:25:47.299",

"minq": "0"

}

]

}

1. 统计表-小时表，表名为：hour\_yyyyMM

原始数据表

* 每个月保存一张小时表，表名规则：hour\_yyyyMM，如hour\_201507。
* 每1分钟更新一次小时表，所以不能当做实时数据进行访问
* 每个表的总数据量：10W点\*30天\*24小时=7200万个数据
* 优化考虑：**每天一条记录**，该记录合并了24个小时，所以每个记录最多24个数据。每个表的记录条数：10W点\*30=300W条记录
* 每个数据存放最大、最小、初始值、时间、质量等属性

表hour \_201704记录示例：

{

"n": "Obj.Tag1",

"t": "2017-04-17",

"vs": [

{

"t": "2017-04-17 16",

"fv": "1",

"ft": "2017-04-17 16:25:46.299",

"fq": "0",

"maxv": "1.2",

"maxt": "2017-04-17 16:25:47.299",

"maxq": "0",

"minv": "1.2",

"mint": "2017-04-17 16:25:47.299",

"minq": "0"

}

]

}

1. 统计表-日表，表名为：day

原始数据表

* 所有10年数据存放在一张表
* 每小时更新一次day，所以不能当做实时数据进行访问
* 该表总数据量：10W点\*10年\*360天=3.6亿个数据
* 优化：每月30个数据合并在存放1条记录
* 总记录条数：10W点\*10\*12年=1200万个记录
* 每个数据存放最大、最小、初始值、时间、质量等属性

表day记录示例：

{

"n": "Obj.Tag1",

"t": "2017",

"vs": [

{

"t": "2017-04-17 16",

"fv": "1",

"ft": "2017-04-17 16:25:46.299",

"fq": "0",

"maxv": "1.2",

"maxt": "2017-04-17 16:25:47.299",

"maxq": "0",

"minv": "1.2",

"mint": "2017-04-17 16:25:47.299",

"minq": "0"

}

]

}

1. 统计表-月表，表名为：month

原始数据表

* 所有时间段的月数据存储在一个数据表month
* 每小时更新一次month表，所以不能当做实时数据进行访问
* 该表的总数据量：10W点\*10年\*12=1200万
* 优化：每年12个数据合并在存放1条记录
* 总记录条数：10W点\*10年=100万个记录
* 每个数据存放最大、最小、初始值、时间、质量等属性

表month记录示例：

{

"n": "Obj.Tag1",

"t": "2017",

"fv": "1",

"ft": "2017-04-17 16:25:46.299",

"fq": "0",

"maxv": "1.2",

"maxt": "2017-04-17 16:25:47.299",

"maxq": "0",

"minv": "1.2",

"mint": "2017-04-17 16:25:47.299",

"minq": "0"

]

}

1. 统计表-年表，表名为：year

* 所有时间段的年数据存储在一个数据表
* 每小时更新一次year表，所以不能当做实时数据进行访问
* 该表总数据量：10W点\*10年\*12=1200万
* 每年12个数据合并在存放1条记录
* 总记录条数：10W点\*10年=100万个记录
* 每个数据存放最大、最小、初始值、时间、质量等属性

表year记录示例：

{

"n": "Obj.Tag1",

"t": "2017",

"fv": "1",

"ft": "2017-04-17 16:25:46.299",

"fq": "0",

"maxv": "1.2",

"maxt": "2017-04-17 16:25:47.299",

"maxq": "0",

"minv": "1.2",

"mint": "2017-04-17 16:25:47.299",

"minq": "0"

}

## 类设计

每个点每张表都需要保存一个最近一次插入后更新时间，作为后续是否插入该表的依据。刚开始为0

Class CTagUpTime{

String m\_strUpTimeTemp;

String m\_strUpTimeMinute;

String m\_strUpTimeHour;

String m\_strUpTimeMonth;

String m\_strUpTimeDay;

String m\_strUpTimeYear;

}

Std::map<string, CTagUpTime \*> g\_mapTagName2UpTime;

每个数据来临时，都需要执行该数据的TAG点的TempData2Raw

TempData2Raw(CTagUpTime \*)

实现：

如果某个点当前时间和Raw表保存的上次时间（2017-05-05 14:20）在十分钟级别（2017-05-05 14:20）不同了，从raw\_20170505读取t为2017-05-05 14:20记录。如果有则更新，没有则插入

TempData2Raw（CTagUpTime \*）执行的最后，调用Raw2Minute

Raw2Minute(CTagUpTime \*)

实现：

如果某个点当前时间和Minute表保存的上次时间（2017-05-05 14）在小时级别（2017-05-05 14）不同了，从minute\_201705读取t为2017-05-05 14记录。如果有则合并现在数据、更新，没有则插入现在的数据。

Raw2Minute(CTagUpTime \*)的最后调用Minute2Hour

Minute2Hour(CTagUpTime \*) 的最后调用Hour2Day

实现：

如果某个点当前时间和Minute表保存的上次时间（2017-05-05 14）在小时级别（2017-05-05 14）不同了，从minute\_201705读取t为2017-05-05 14记录。如果有则更新，没有则插入。

Minute2Hour(CTagUpTime \*) 的最后调用Hour2Day

Hour2Day(CTagUpTime \*) 的最后调用Day2Month

Day2Month(CTagUpTime \*) 的最后调用Month2Year

Month2Year(CTagUpTime \*) 的最后调用Minute2Hour

启动时，需要读取Temp表，每个tag点执行一次TempData2Raw(CTagUpTime \*)